URL-Based Phishing Detection Using Machine Learning

INTRODUCTION

1.1 Overview

There are a number of users who purchase products online and make payments through e-banking. There are e-banking websites that ask users to provide sensitive data such as username, password & credit card details, etc., often for malicious reasons. This type of e-banking website is known as a phishing website. Web service is one of the key communications software services for the Internet. Web phishing is one of many security threats to web services on the Internet.

Web phishing aims to steal private information, such as usernames, passwords, and credit card details, by way of impersonating a legitimate entity. It will lead to information disclosure and property damage. Large organizations may get trapped in different kinds of scams.

This Project mainly focuses on applying a machine-learning algorithm to detect Phishing websites.

In order to detect and predict e-banking phishing websites, we proposed an intelligent, flexible and effective system that is based on using classification algorithms. We implemented classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy. The e-banking phishing website can be detected based on some important characteristics like URL and domain identity, and security and encryption criteria in the final phishing detection rate. Once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

1.2 Purpose

The main purpose of the project is to detect the fake or phishing websites who are trying to get access to the sensitive data or by creating the fake websites and trying to get access of the user personal credentials. We are using machine learning algorithms to safeguard the sensitive data and to detect the phishing websites who are trying to gain access on sensitive data.

It can be used to preserve the confidentiality. To protect the user from phishing websites. To develop a user-friendly environment. To prevent or mitigate harm or destruction of computer networks, applications, devices, and data.

LITERATURE SURVEY

2.1 Existing problem

The existing system handles the only one kind of phishing attacks. If that was a phishing site then the existing system only warns the user. The active warning gives the user options to close the window or displaying the website. The passive warning displays the popup dialog box.

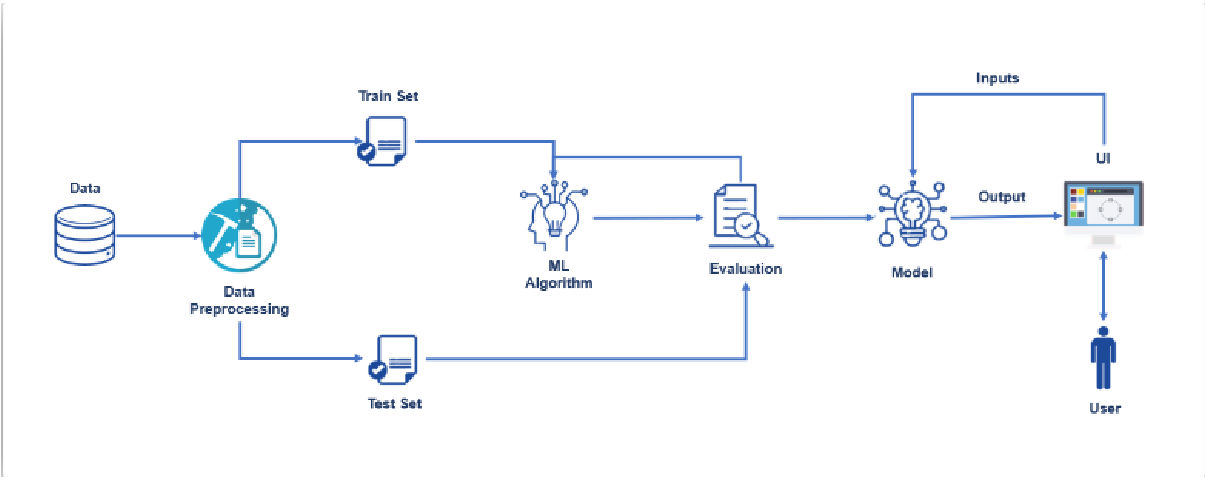
2.2 Proposed solution

The dataset of phishing and legitimate URL's is given to the system which is then pre-processed so that the data is in the useable format for analysis. The features have around 30 characteristics of phishing websites which is used to differentiate it from legitimate ones. Each category has its own characteristics of phishing attributes and values are defined. The specified characteristics are extracted for each URL and valid ranges of inputs are identified. These values are then assigned to each phishing website risk. For each input the values range from 0 to 10 , while for output range is from 0 to 100. The phishing attributes values are represented with binary no 0 and 1 which indicates the attribute is present or not.

After this the data is trained we shall apply a relevant machine learning algorithm to the dataset. The machine learning algorithms are already explained in previous section. After this we use a hybrid classification in which we combine two of the classifier namely Naive Bayes and Random forest to predict the accuracy of the detection of the phishing URL, hence we get our desired result. This is also called a hybrid approach to test the data, in this method we propose to use the combination of two classifiers, as mentioned above. We shall then test the data and evaluate the prediction accuracy which shall be more than the existing system. We shall now see the different classifiers and discuss the hybrid combination used for our proposed system.

THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Hardware / Software designing

Hardware Requirements:

The following is the hardware requirements of the system for the proposed system:

• Processor	: Any Processor above 500 MHz
• RAM	:8 GB
• Hard Disk	:1 TB
• Input device	: Standard keyboard and mouse

Software Requirements:

The following is the software requirements of the system for the proposed system:

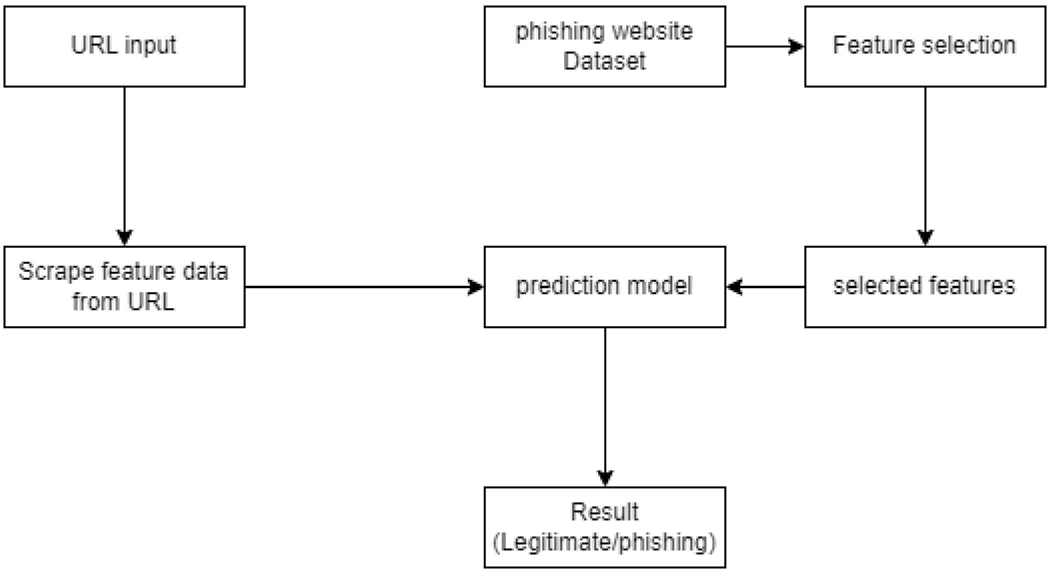
• OS	: Windows 10
• Platform	: Jupyter Notebook
• Language	: Python
• IDE/tool	: Anaconda 3-5.0.3

EXPERIMENTAL INVESTIGATIONS

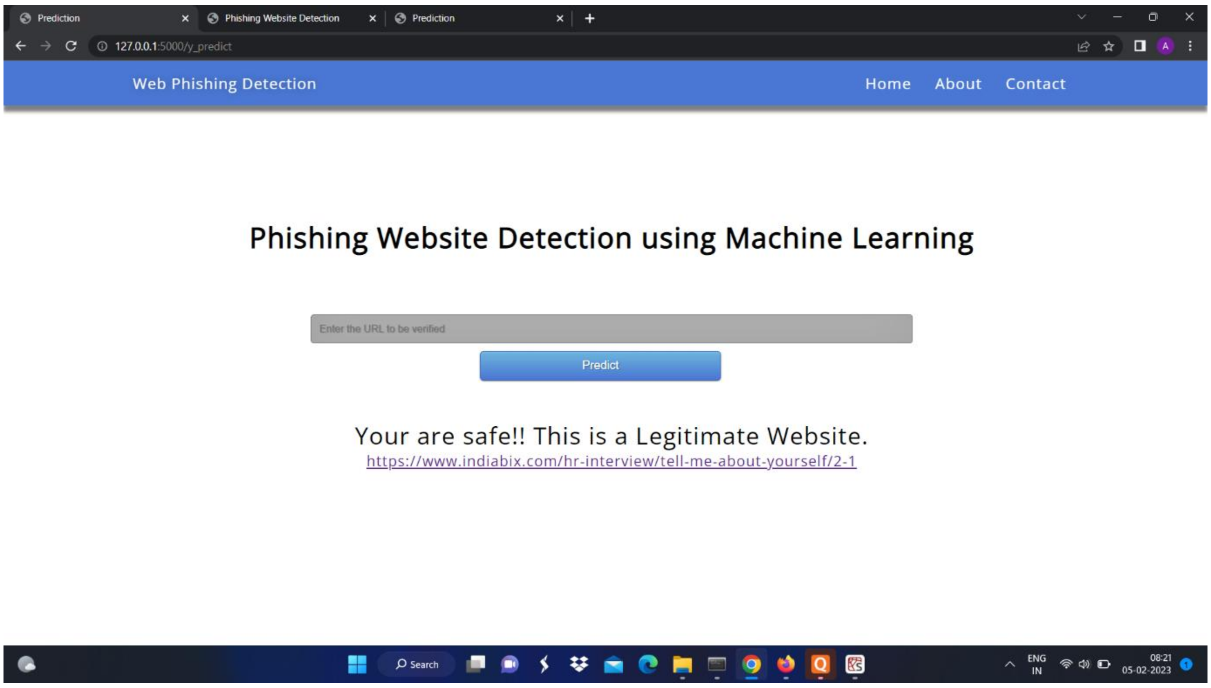
Scikit-learn tool has been used to import Machine Learning algorithms. Dataset is divided into training set and testing set in 70:30 ratio. Each classifier is trained using training data and testing data which is used to analyse the performance of classifier. Performance of classifier has been evaluated by calculating classifier's accuracy score.

Decision Tree	: 0.9461782
Random Forest	: 0.9104480
SVM	: 0.541384

FLOWCHART



RESULT



Phishing Website Detection using Machine Learning

Your are safe!! This is a Legitimate Website.

https://www.indiabix.com/hr-interview/tell-me-about-yourself/2-1

ADVANTAGES & DISADVANTAGES

Advantages

- This system can be used by many E-commerce or other websites in order to have good customer relationship.
- User can make online payment securely.
- Data mining algorithm used in this system provides better performance as compared to other traditional classifications algorithms.
- With the help of this system user can also purchase products online without any hesitation.

Disadvantages

- If Internet connection fails, this system won’t work.
- All websites related data will be stored in one place.

APPLICATIONS

Phishing is one of the most severe cyber-attacks where researchers are interested to find a solution. In phishing, attackers lure end-users and steal their personal in-formation. To minimize the damage caused by phishing must be detected as early as possible. There are various phishing attacks like spear phishing, whaling, vishing, smishing, pharming and so on. There are various phishing detection techniques based on white-list, black-list, content-based, URL-based, visual-similarity and machine-learning.

The importance to safeguard online users from becoming victims of online fraud, divulging confidential information to an attacker among other effective uses of phishing as an attacker's tool, phishing detection tools play a vital role in ensuring a secure online experience for users.

CONCLUSION

The demonstration of phishing is turning into an advanced danger to this quickly developing universe of innovation. Today, every nation is focusing on cashless exchanges, business online, tickets that are paperless and so on to update with the growing world. Yet phishing is turning into an impediment to this advancement. Individuals are not feeling web is dependable now. It is conceivable to utilize AI to get information and assemble extraordinary information items. A lay person, completely unconscious of how to recognize a security danger shall never invite the danger of making money related exchanges on the web. Phishers are focusing on installment industry and cloud benefits the most. The project means to investigate this region by indicating an utilization instance of recognizing phishing sites utilizing ML. It aimed to build a phishing detection mechanism using machine learning tools and techniques which is efficient, accurate and cost effective. The project was carried out in Anaconda IDE and was written in Python. The proposed method used four machine learning classifiers to achieve this and a comparative study of the four algorithms was made. A good accuracy score was also achieved. All the three classifiers gave promising results with the best being Decision Tree Classifier with an accuracy score of 95.6%. The accuracy score might vary while using other datasets and other algorithms might provide better accuracy than decision tree classifier. This model can be deployed in real time to detect the URLs as phishing or legitimate

FUTURE SCOPE

The proposed idea can be improvised by detecting the clickable pictures, malicious QR code, etc. The limitation is that all features are discrete. The other limitation is that the URL is to be copied and we have to search in the application then it will predict whether it is legitimate or not rather than redirecting the URL link to the application. If the URL is not there in the training and testing data set then it is difficult to predict that the URL is legitimate or not.

Further work can be done to enhance the model by using ensembling models to get greater accuracy score. Ensemble methods is a ML technique that combines many base models to generate an optimal predictive model. Further reaching future work would be combining multiple classifiers, trained on different aspects of the same training set, into a single classifier that may provide a more robust prediction than any of the single classifiers on their own. The project can also include other variants of phishing like smishing, vishing, etc. to complete the system. Looking even further out, the methodology needs to be evaluated on how it might handle collection growth. The collections will ideally grow incrementally over time so there will need to be a way to apply a classifier incrementally to the new data, but also potentially have this classifier receive feedback that might modify it over time.

BIBILOGRAPHY

[1] Github. Github. https://github.com/

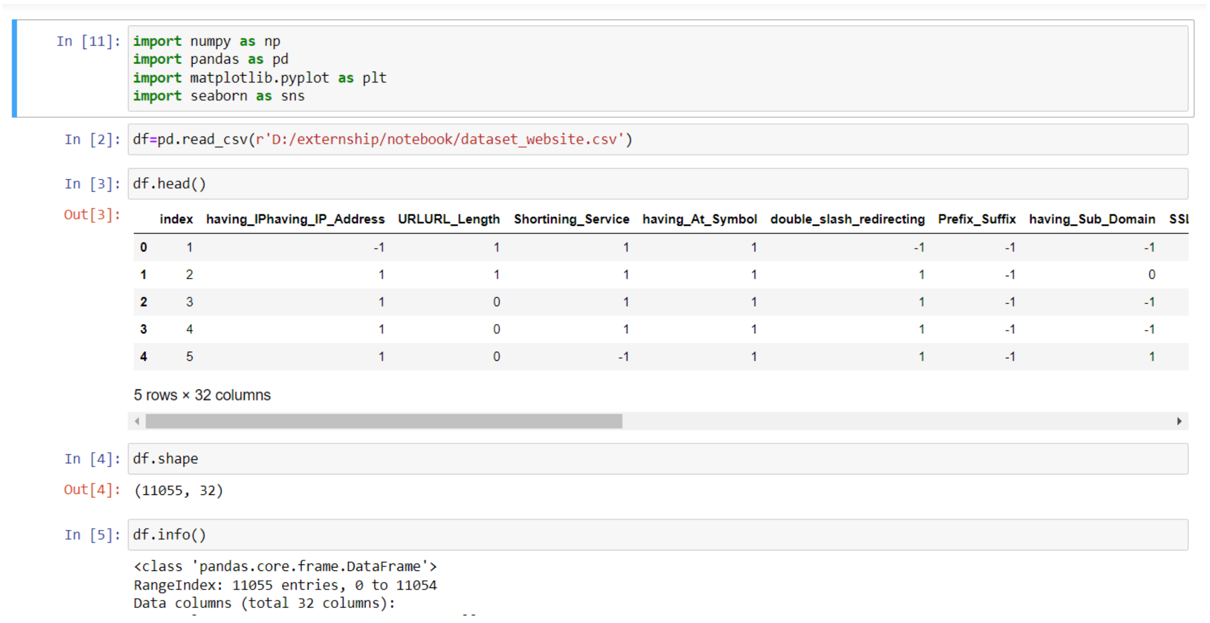
[2] SmartInternz student portal

[3] YouTube

APPENDIX

A. Source Code

Jupyter notebook



In [11]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```python
df=pd.read_csv(r'D:/externship/notebook/dataset_website.csv')
```

In [3]:
```python
df.head()
```

Out[3]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSl
0	1	-1	1	1	1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	
2	3	1	0	1	1	1	-1	-1	
3	4	1	0	1	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	

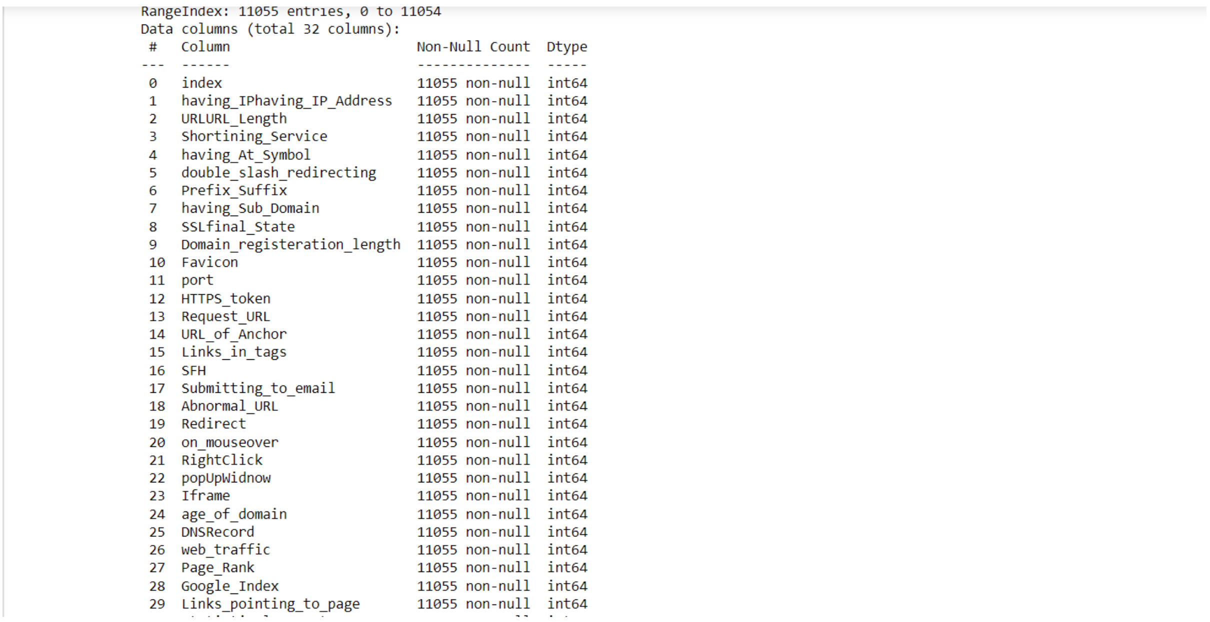
5 rows × 32 columns

In [4]:
```python
df.shape
```

Out[4]:
```
(11055, 32)
```

In [5]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 32 columns):
```



```
RangeIndex: 11055 entries, 0 to 11054
Data columns (total 32 columns):
```

```
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   index                       11055 non-null  int64
 1   having_IPhaving_IP_Address  11055 non-null  int64
 2   URLURL_Length               11055 non-null  int64
 3   Shortining_Service          11055 non-null  int64
 4   having_At_Symbol            11055 non-null  int64
 5   double_slash_redirecting    11055 non-null  int64
 6   Prefix_Suffix               11055 non-null  int64
 7   having_Sub_Domain           11055 non-null  int64
 8   SSLfinal_State              11055 non-null  int64
 9   Domain_registeration_length 11055 non-null  int64
 10  Favicon                     11055 non-null  int64
 11  port                        11055 non-null  int64
 12  HTTPS_token                 11055 non-null  int64
 13  Request_URL                 11055 non-null  int64
 14  URL_of_Anchor               11055 non-null  int64
 15  Links_in_tags               11055 non-null  int64
 16  SFH                         11055 non-null  int64
 17  Submitting_to_email         11055 non-null  int64
 18  Abnormal_URL                11055 non-null  int64
 19  Redirect                    11055 non-null  int64
 20  on_mouseover                11055 non-null  int64
 21  RightClick                  11055 non-null  int64
 22  popUpWidnow                 11055 non-null  int64
 23  Iframe                      11055 non-null  int64
 24  age_of_domain               11055 non-null  int64
 25  DNSRecord                   11055 non-null  int64
 26  web_traffic                 11055 non-null  int64
 27  Page_Rank                   11055 non-null  int64
 28  Google_Index                11055 non-null  int64
 29  Links_pointing_to_page      11055 non-null  int64
 30  Statistical_report           11055 non-null  int64
 31  Result                       11055 non-null  int64
dtypes: int64(32)
memory usage: 2.7 MB
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
index                          0
having_IPhaving_IP_Address     0
URLURL_Length                  0
Shortining_Service             0
having_At_Symbol               0
double_slash_redirecting       0
Prefix_Suffix                  0
having_Sub_Domain              0
SSLfinal_State                 0
Domain_registeration_length    0
Favicon                        0
port                           0
HTTPS_token                    0
Request_URL                    0
URL_of_Anchor                  0
Links_in_tags                  0
SFH                            0
Submitting_to_email            0
Abnormal_URL                   0
Redirect                       0
on_mouseover                   0
RightClick                     0
popUpWidnow                    0
Iframe                         0
age_of_domain                  0
DNSRecord                      0
web_traffic                    0
```

In [7]:
```
df.describe()
```

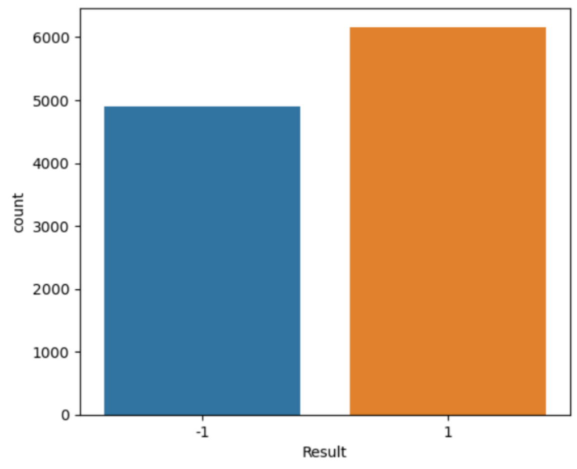
Out[7]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_I
count	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.000000	11055.
mean	5528.000000	0.313795	-0.633198	0.738761	0.700588	0.741474	-0.734962	0
std	3191.447947	0.949534	0.766095	0.673998	0.713598	0.671011	0.678139	0
min	1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1
25%	2764.500000	-1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	-1
50%	5528.000000	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	0
75%	8291.500000	1.000000	-1.000000	1.000000	1.000000	1.000000	-1.000000	1
max	11055.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1

8 rows × 32 columns

In [12]:
```
plt.figure(figsize=(6,5))
sns.countplot(df['Result'])
```

```
C:\Users\Hp\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg:
x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```



In [13]:
```
df.corr()
```

Out[13]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Su
index	1.000000	-0.388317	0.006105	-0.006281	-0.169478	-0.003363	-0.007

having_IPhaving_IP_Address	-0.388317	1.000000	-0.052411	0.403461	0.158699	0.397389	-0.005
URLURL_Length	0.006105	-0.052411	1.000000	-0.097881	-0.075108	-0.081247	0.055
Shortining_Service	-0.006281	0.403461	-0.097881	1.000000	0.104447	0.842796	-0.080
having_At_Symbol	-0.169478	0.158699	-0.075108	0.104447	1.000000	0.086960	-0.011
double_slash_redirecting	-0.003363	0.397389	-0.081247	0.842796	0.086960	1.000000	-0.085
Prefix_Suffix	-0.007340	-0.005257	0.055247	-0.080471	-0.011726	-0.085590	1.000
having_Sub_Domain	0.234091	-0.080745	0.003997	-0.041916	-0.058976	-0.043079	0.087
SSLfinal_State	-0.006682	0.071414	0.048754	-0.061426	0.031220	-0.036200	0.261
Domain_registeration_length	-0.001180	-0.022739	-0.221892	0.060923	0.015522	0.047464	-0.096
Favicon	0.007293	0.087025	-0.042497	0.006101	0.304899	0.035100	-0.007
port	0.001656	0.060979	0.000323	0.002201	0.364891	0.025060	-0.022
HTTPS_token	0.002916	0.363534	-0.089383	0.757838	0.104561	0.760799	-0.070
Request_URL	-0.000862	0.029773	0.246348	-0.037235	0.027909	-0.026368	0.098
URL_of_Anchor	-0.005071	0.099847	-0.023396	0.000561	0.057914	-0.005036	0.348
Links_in_tags	-0.028865	0.006212	0.052869	-0.133379	-0.070861	-0.125583	0.100
SFH	0.085354	-0.010962	0.414196	-0.022723	-0.008672	-0.041672	0.001
Submitting_to_email	0.005828	0.077989	-0.014457	0.049328	0.370123	0.031898	-0.045
Abnormal_URL	0.003228	0.336549	-0.106761	0.739290	0.203945	0.723724	-0.077
Redirect	0.016804	-0.321181	0.046832	-0.534530	-0.028160	-0.591478	0.016
on_mouseover	0.003649	0.084059	-0.045103	0.062383	0.279697	0.086635	0.012

In [14]:
```
x=df.drop(columns=['Result'])
x.head()
```

Out[14]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain	SSl
0	1	-1	1	1	1	-1	-1	-1	
1	2	1	1	1	1	1	-1	0	
2	3	1	0	1	1	1	-1	-1	
3	4	1	0	1	1	1	-1	-1	
4	5	1	0	-1	1	1	-1	1	

5 rows × 31 columns

In [15]:
```
y=df['Result']
y.head()
```

Out[15]:
```
0   -1
1   -1
2   -1
3   -1
4    1
Name: Result, dtype: int64
```

In [16]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [17]:
```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion='entropy')
dtc.fit(x train,y train)
```

```
dtc.fit(x_train,y_train)
```

Out[17]:
```
DecisionTreeClassifier(criterion='entropy')
```

In [18]:
```
x_test
```

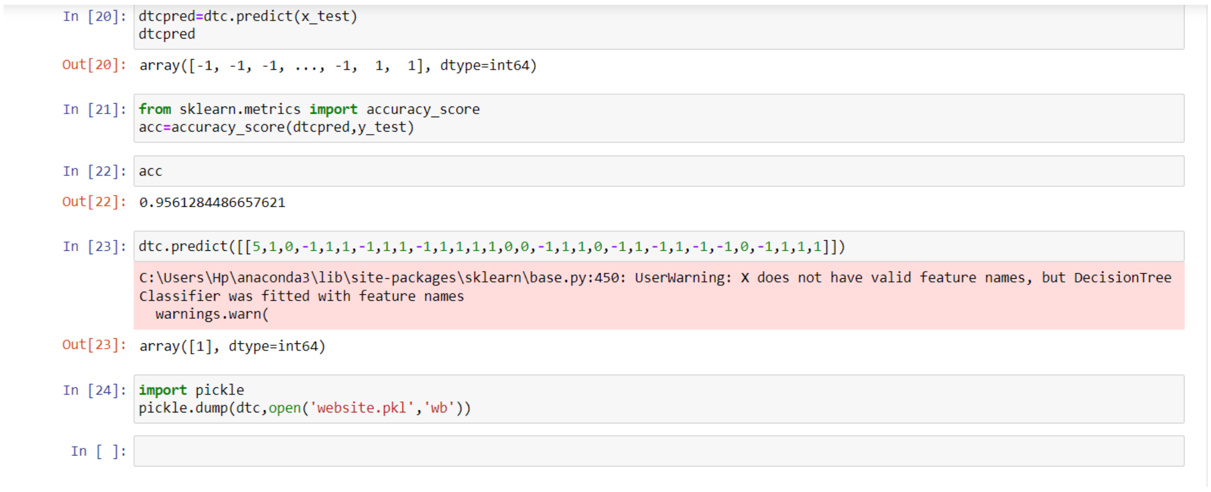
Out[18]:

	index	having_IPhaving_IP_Address	URLURL_Length	Shortining_Service	having_At_Symbol	double_slash_redirecting	Prefix_Suffix	having_Sub_Domain
226	227	1	-1	1	1	1	-1	-1
2252	2253	1	-1	1	1	1	-1	0
2646	2647	1	-1	1	1	1	-1	0
6444	6445	1	-1	1	1	1	-1	0
1387	1388	1	-1	1	1	1	-1	0
...	...	...	...	...	...	...	...	...
5831	5832	-1	-1	1	1	1	-1	0
8541	8542	-1	-1	1	-1	1	-1	-1
6810	6811	-1	-1	1	-1	1	-1	1
4721	4722	1	-1	1	1	1	-1	0
325	326	1	-1	1	1	1	1	-1

2211 rows × 31 columns

In [19]:
```
y_test
```

Out[19]:
```
226     -1
2252    -1
2646     1
```



```
In [20]: dtcpred=dtc.predict(x_test)
         dtcpred

Out[20]: array([-1, -1, -1, ..., -1,  1,  1], dtype=int64)

In [21]: from sklearn.metrics import accuracy_score
         acc=accuracy_score(dtcpred,y_test)

In [22]: acc

Out[22]: 0.9561284486657621

In [23]: dtc.predict([[5,1,0,-1,1,1,-1,1,1,-1,1,1,1,1,0,0,-1,1,1,0,-1,1,-1,1,-1,-1,0,-1,1,1,1]])

         C:\Users\Hp\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTree
         Classifier was fitted with feature names
           warnings.warn(

Out[23]: array([1], dtype=int64)

In [24]: import pickle
         pickle.dump(dtc,open('website.pkl','wb'))

In [ ]:
```

app.py

```
import numpy as np

from flask import Flask, request, jsonify, render_template

import pickle

#importing the inputScript file used to analyze the URL

import inputScript

#load model

app = Flask(__name__)

model = pickle.load(open('website.pkl', 'rb'))


#Redirects to the page to give the user iput URL.

@app.route('/')

def home():

    return render_template('index.html')

@app.route('/predict')

def predict():

    return render_template('final.html')


#Fetches the URL given by the URL and passes to inputScript

@app.route('/y_predict',methods=['POST'])

def y_predict():

    '''
    For rendering results on HTML GUI
    '''
    url = request.form['URL']
    checkprediction = inputScript.main(url)
    prediction = model.predict(checkprediction)
    print(prediction)
    output=prediction[0]
    if(output==1):
        pred="Your are safe!!  This is a Legitimate Website."

    else:
        pred="You are on the wrong site. Be cautious!"
    return render_template('final.html', prediction_text='{}'.format(pred),url=url)

#Takes the input parameters fetched from the URL by inputScript and returns the predictions
@app.route('/predict_api',methods=['POST'])
def predict_api():
    '''
    For direct API calls trought request
    '''
    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

if __name__ == "__main__":
    app.run(debug=True)
if __name__ == '__main__':
    app.run(host='0.0.0.0', debug=True)
```